Aufgabe 1: 3-Schichten-Architektur eines DBS (14 Punkte)

Die 3 Datenebenen heißen: (1 Punkt)
- externes Modell
- konzeptuelles Modell
- internes Modell

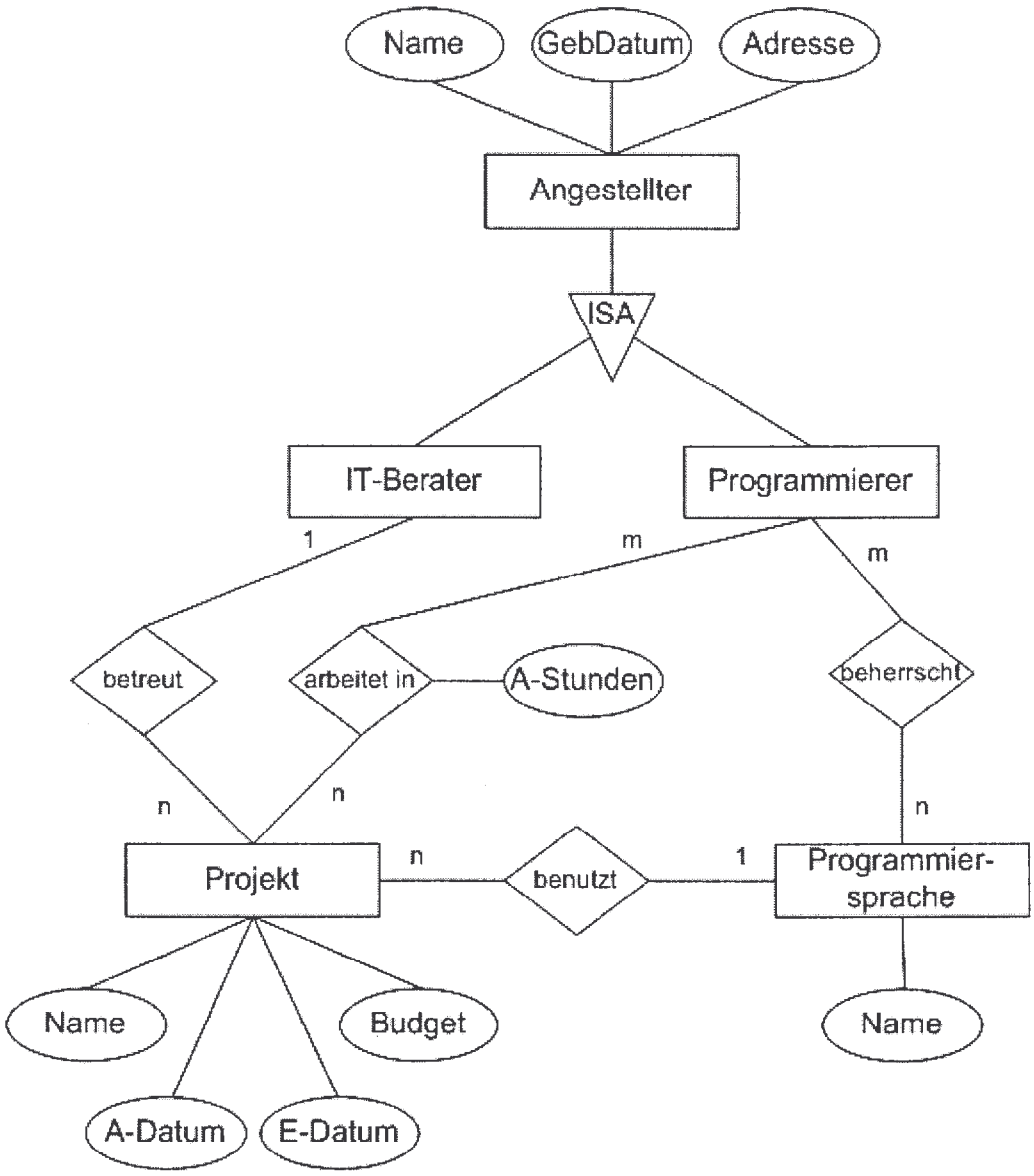
a) Das externe Modell: (3 Punkte)
Das externe Modell beschreibt die einzelnen Sichten der Benutzer. Jeder Benutzer soll nur die für ihn relevanten Daten sehen. Die Objekttypen und Beziehungstypen müssen nicht mit denen des konzeptuellen Modells identisch sein, sie müssen lediglich „inhaltlich" im konzeptuellen Modell enthalten sein. (2 Punkte)
Die Erstellung des entsprechenden Schemas hängt von den Anforderungen und auch von den Befugnissen (Zugriffsrechten usw.) des jeweiligen Benutzers ab. (1 Punkt)

b) Das konzeptuelle Modell: (3 Punkte)
Das konzeptuelle Modell beschreibt die Gesamtheit aller Daten, die innerhalb der Datenbank verwaltet werden (1 Punkt). Dazu gehören die Festlegungen von Beziehungen, Integritätsbedingungen und Operationen (1 Punkt).
Das zu erstellende Schema hängt einzig und allein von den in der Realität existierenden Strukturen (die wesentlichen Daten eines Unternehmens und die wesentlichen Beziehungen zwischen den Daten) ab (1 Punkt).

c) Das interne Modell: (4 Punkte)
Das interne Modell beschreibt die vom Datenbankadministrator festgelegte physische Datenorganisation (1 Punkt). Es enthält u.a. alle Informationen über den Aufbau der abgespeicherten Daten, die Speicherung der Daten und über die Zugriffspfade (1 Punkt). Es muß genaue Festlegungen der folgenden Punkte enthalten (1 Punkt):
- Repräsentation von Attributwerten
- Aufbau gespeicherter Sätze
- Zugriffsmethoden auf Sätze
- zusätzliche Zugriffspfade (Indizes, Verkettungen, usw.)
- Die Erstellung des internen Schemas hängt von den statischen Informationen über die Häufigkeit von Anwendungen, von Zugriffen auf Objekten, über Zeitbeschränkungen für Anwendungen, Verteilung von Werten für Attribute usw. ab (1 Punkt).

Datenunabhängigkeit: (3 Punkte)
Datenunabhängigkeit bedeutet, daß Anwendungsprogramme von Änderungen auf der internen und der konzeptuellen Ebene unberührt bleiben. Man unterscheidet physische und logische Datenunabhängigkeit. (1 Punkt)
Physische Datenunabhängigkeit bedeutet Isolierung der Anwendungsprogramme vor Änderungen der physischen Datenorganisation. (1 Punkt)
Logische Datenunabhängigkeit bedeutet Isolierung der Anwendungsprogramme vor Änderungen des konzeptuellen Modells. (1 Punkt)

Aufgabe 2: Entity-Relationship-Diagramm (11 Punkte)



Aufgabe 3: Datenmodellierung (14 Punkte)

a) Jeweils 1 Punkt:
Plattenfirma = (Name, Adresse)
Album = (a_id, Titel, PlattenfirmaName, Datum)
SoloAlbum = (a_id, aufgenommenVon)
BandAlbum = (a_id, aufgenommenVon)
Künstler = (k_id, Name, E-Mail)
Band = (b_id, Name, Bandleader)
TeilVon = (k_id, b_id)

Alternativ:
nurAlbum = (a_id, Titel, PlattenfirmaName, Datum)
SoloAlbum = (a_id, Titel, PlattenfirmaName, Datum, aufgenommenVon)
BandAlbum = a_id, Titel, PlattenfirmaName, Datum, aufgenommenVon)

b) Jeweils 1 Punkt:

```
CREATE TABLE Plattenfirma
        (Name CHAR(50) NOT NULL,
        Adresse CHAR(50),
        PRIMARY KEY (Name))

CREATE TABLE Album
        (a_id INTEGER NOT NULL,
        Titel CHAR(50),
        PlattenfirmaName CHAR(50) NOT NULL,
        Datum date,
        PRIMARY KEY (a_id),
        FOREIGN KEY (PlattenfirmaName) REFERENCES Plattenfirma)

CREATE TABLE Soloalbum
        (a_id INTEGER NOT NULL,
        aufgenommenVon INTEGER NOT NULL,
        PRIMARY KEY (a_id),
        FOREIGN KEY (aufgenommenVon) REFERENCES Künstler)

CREATE TABLE Bandalbum
        (a_id INTEGER NOT NULL,
        aufgenommenVon INTEGER NOT NULL,
        PRIMARY KEY (a_id),
        FOREIGN KEY (aufgenommenVon) REFERENCES Band)

CREATE TABLE Künstler
        (k_id INTEGER NOT NULL,
        Name CHAR(50),
        E-Mail CHAR(50),
        PRIMARY KEY (k_id))

CREATE TABLE Band
        (b_id INTEGER NOT NULL,
        Name CHAR(50),
        Bandleader INTEGER NOT NULL,
        PRIMARY KEY (b_id),
        FOREIGN KEY (Bandleader) REFERENCES Künstler)

CREATE TABLE TeilVon
        (k_id INTEGER NOT NULL,
        b_id INTEGER NOT NULL,
        PRIMARY KEY (k_id, b_id),
        FOREIGN KEY (k_id) REFERENCES Künstler,
        FOREIGN KEY (b_id) REFERENCES Band)
```

Alternativ:

```
CREATE TABLE nurAlbum
        (a_id INTEGER NOT NULL,
        Titel CHAR(50),
        PlattenfirmaName CHAR(50) NOT NULL,
        Datum date,
        PRIMARY KEY (a_id),
        FOREIGN KEY (PlattenfirmaName) REFERENCES Plattenfirma)

CREATE TABLE Soloalbum
        (a_id INTEGER NOT NULL,
        Titel CHAR(50),
        PlattenfirmaName CHAR(50) NOT NULL,
        Datum date,
        aufgenommenVon INTEGER NOT NULL,
        PRIMARY KEY (a_id),
        FOREIGN KEY (PlattenfirmaName) REFERENCES Plattenfirma,
        FOREIGN KEY (aufgenommenVon) REFERENCES Künstler)

CREATE TABLE Bandalbum
        (a_id INTEGER NOT NULL,
        Titel CHAR(50),
        PlattenfirmaName CHAR(50) NOT NULL,
        Datum date,
        aufgenommenVon INTEGER NOT NULL,
        PRIMARY KEY (a_id),
        FOREIGN KEY (PlattenfirmaName) REFERENCES Plattenfirma,
        FOREIGN KEY (aufgenommenVon) REFERENCES Band)
```

Aufgabe 4: Queries **(19 Punkte)**

a) (3 Punkte)

```
SELECT Autor
FROM Buch
WHERE Verlag =   SELECT Verlag
                 FROM Buch
                 WHERE Autor = ´G. Schlageter´
                 AND Titel = ´Datenbanksysteme´
```

b) (4 Punkte)

```
SELECT Name, Anschrift
FROM Entleiher e
WHERE 20 <       SELECT COUNT(BuchNr)
                 FROM Buch_Ent
                 WHERE EntleiherNr = e.EntleiherNr
```

oder

```
SELECT Name, Anschrift
FROM Entleiher
WHERE EntleiherNr IN    (SELECT EntleiherNr
                         FROM Buch_Ent
                         WHERE COUNT(BuchNr) > 20
                         GROUP BY EntleiherNr)
```

c) (3 Punkte)
Buch [Autor = ´G. Schlageter´] [BuchNr = BuchNr] Buch_Ent
[Entleihdatum > 1.10.2004] [EntleihNr = EntleihNr] Entleiher
[Name, Anschrift]

d) (2 Punkte)

```
CREATE VIEW Zahl_der_Bücher (Standort, Anzahl) AS
        SELECT Standort COUNT(BuchNr)
        FROM Buch
        GROUP BY Standort
```

e) (3 Punkte)
RANGE Buch X
RANGE Buch_Ent Y
{ Entleiher.Name, Entleiher.Anschrift |
¬ ∃ Y (Y.EntleihNr = Entleiher.EntleihNr ∧ Y. Entleihdatum > 31.12.2004 ∧
∃ X (X.BuchNr = Y.BuchNr ∧ X.Verlag = ´Addison-Wesley´)) }

f) (4 Punkte)
RANGE Buch_Ent X
RANGE Buch_Ent Y
{ Entleiher.Name, Entleiher.Anschrift |
∃ X (X.EntleiherNr = Entleiher.EntleiherNr ∧
¬ ∃ Y (Y.EntleiherNr = X.EntleiherNr ∧ Y.Entleihdatum <> X.Entleihdatum)) }

Aufgabe 5: Funktionale Abhängigkeiten, Normalformen (10 Punkte)

a) Die Relation befindet sich in erster Normalform, da die Wertebereiche jedes Attributs unteilbare Werte enthalten und nicht ihrerseits wieder aus Mengen oder Tupeln bestehen. (1 Punkt)

Für die 2. Normalform muss zunächst der Schüssel ermittelt werden. Keines der Attribute allein bildet einen Schlüssel, aber von {Titel, Autor} sind alle anderen Attribute über die funktionalen Abhängigkeiten erreichbar. Damit ist {Titel, Autor} ein Schlüsselkandidat.

Die Relation befindet sich nicht in 2. Normalform, weil Attribute von Teilen des Schlüssels abhängig sind. (1 Punkt)

Die Relation befindet sich nicht in 3. Normalform, weil es funktionale Abhängigkeiten gibt, die nicht von einem Schlüsselattribut ausgehen (Buchtyp → Listenpreis). (1 Punkt)

b) Um die Relation in die 2. Normalform zu bringen, müssen die Nichtschlüsselattribute voll funktional abhängig vom Schlüssel sein. (1 Punkt)

Daher zerlegen wir wie folgt: (jeweils 1 Punkt)

Buch-Autor (Titel, Autor)
Buch-1 (Titel, Verlag, Buchtyp, Listenpreis)
Autor-1 (Autor, Adresse_Autor)

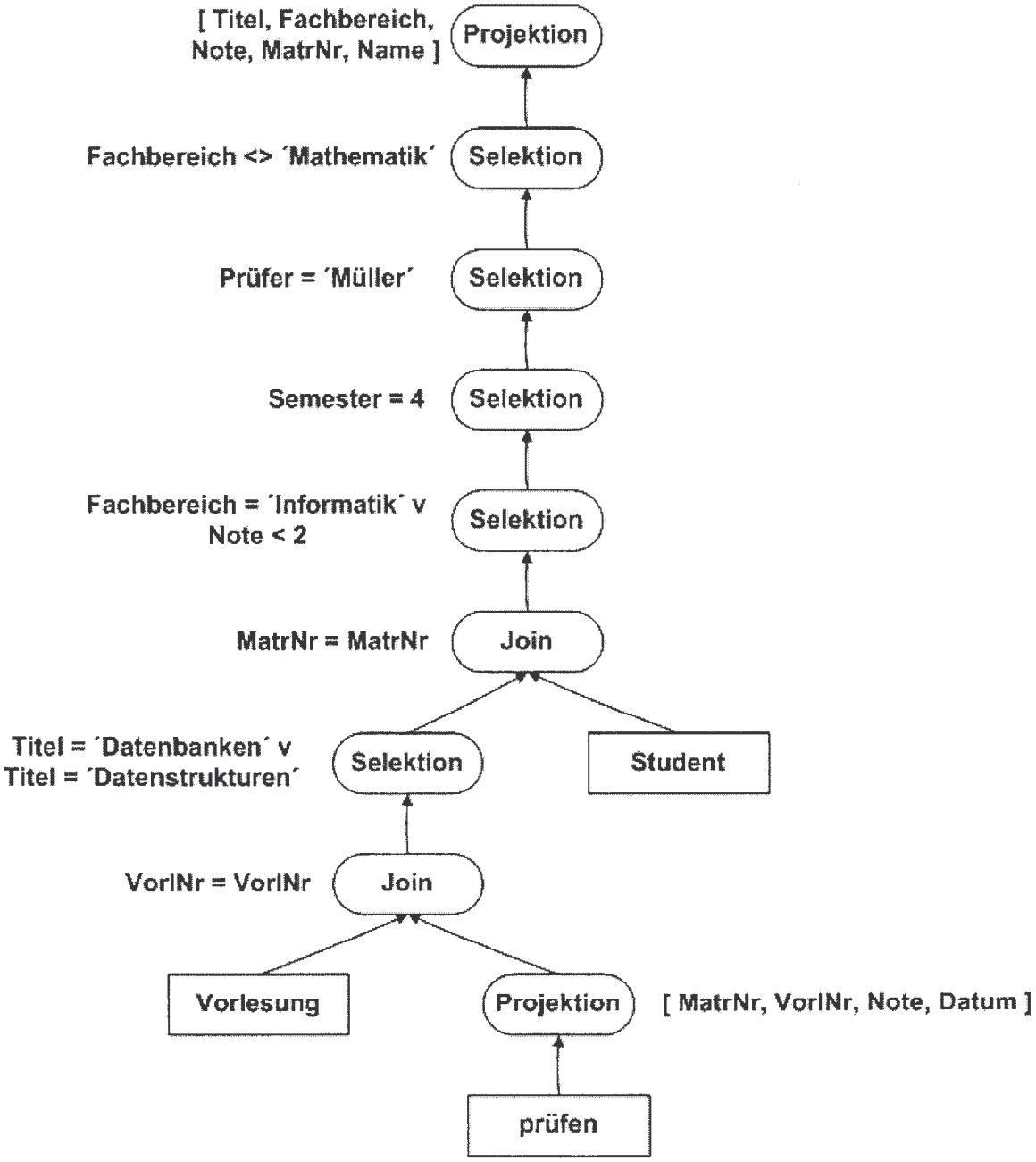
Um das Relationschema in 3. Normalform zu bringen, muß die Abhängigkeit in Buch-1 eliminiert werden, die nicht von einem Schlüsselattribut ausgeht. (1 Punkt)

Daher zerlegen wir wie folgt: (jeweils 1 Punkt)

Buch-2 (Titel, Verlag, Buchtyp)
Buch-Preis (Buchtyp, Listenpreis)

Aufgabe 6: Anfrageoptimierung (14 Punkte)

a) Operatorbaum: (2 Punkte)



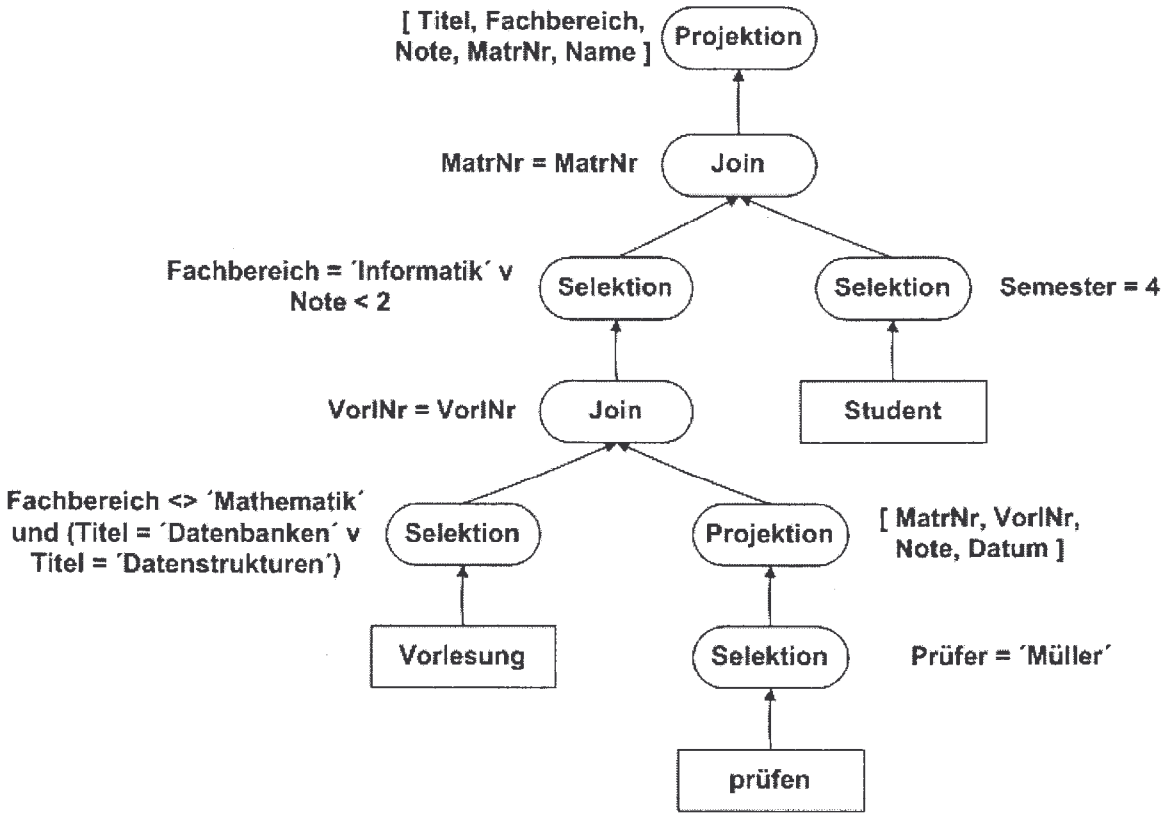
b) Die Selektion **[Prüfer = ´Müller´]** bezeiht sich nur auf die Relation **prüfen** und kann daher vor den Verbünden ausgeführt werden.
[Titel = ´Datenbanken´ v Titel = ´Datenstrukturen´] und **[Fachbereich <> ´Mathematik´]** sind Selektionen, die sich beide auf die Relation **Vorlesungen** beziehen. Deshalb ist es möglich, diese zu einer Selektion zusammenzufassen und sie vor dem Verbund **[VorlNr = VorlNr]** auszuführen.

Die Selektion **[Semester = 4]** bezieht sich nur auf die Relation **Studenten** und kann daher vor dem Verbund **[MatrNr = MatrNr]** ausgeführt werden.
Die Selektion **[Fachbereich = ´Informatik´ v Note < 2]** kann, da es sich um eine ODER-Verknüpfung handelt, nicht aufgespalten werden. Sie bezieht sich jedoch nur auf die beiden Relationen **Vorlesungen** und **prüfen** und kann daher auch vor dem Verbund **[MatrNr = MatrNr]** ausgeführt werden.
(4 Punkte)

Als Resultat der Optimierung ergibt sich folgender Ausdruck: (2 Punkte)

Vorlesung [Fachbereich <> ´Mathematik´ und (Titel = ´Datenbanken´ v
Titel = ´Datenstrukturen´)] [VorlNr = VorlNr] (prüfen [Prüfer = ´Müller´]
[MatrNr, VorlNr, Note, Datum]) [Fachbereich = ´Informatik´ v Note < 2]
[MatrNr = MatrNr] (Studenten [Semester = 4])
[Titel, Fachbereich, Note, MatrNr, Name]

Operatorbaum: (2 Punkte)



c) Die notwendigen Verbund-Operationen (z.B. **R [A = B] S**) lassen sich auf physischer Ebene noch optimieren durch:

Ausnutzung von Sekundärindizes
Existiert ein Index für **B**, dann ist für jedes Tupel in **R** sofort die zugehörige Tupelmenge in **S** feststellbar – und damit die Tupelmenge des Verbundes konstruierbar. Die Zahl der notwendigen Zugriffe auf Tupel wird damit linear zur Größe der beiden Relationen.

*Sortieren von **R** nach **A** und von **S** nach **B***
Bei effizienten Sortierverfahren werden O(n log n) Schritte benötigt. Das anschließende Erstellen des Verbundes ist dann sehr schnell: **R** und **S** werden parallel durchlaufen, wobei jeweils auf gleich Werte für **A** und **B** zu prüfen ist.

Aufgabe 7: Recovery (11 Punkte)

a) Ein physisches Log enthält die Before- und After-Images jeder Update-Operation. Ein logisches Log enthält nur einen Eintrag, der die Update-Operation beschreibt (etwa: „insert record r into file x"). (1 Punkt)

Vorteil: Der Speicheraufwand für das Log und die Belastung der I/O-Kanäle ist geringer, da beim physischen Log für eine einzige Operation u.U. eine ganze Reihe von Before- und After-Images geschrieben werden müssen. (1 Punkt)

Nachteil: Zum einen muß der Recovery-Manager erheblich erweitert werden, so daß er in der Lage ist, nicht nur die entsprechenden Operationen „nachzuführen", sondern auch die inversen Operationen ausführen kann. Zum anderen ist es für die UNDO-Operation wesentlich zu wissen, ob die entsprechende Operation sich bereits auf die Datenbank ausgewirkt hat oder nicht (unerheblich für das UNDO beim physischen Log.) (2 Punkte)

b) Beim Shadow-Verfahren werden alle Änderungen zum Commit-Zeitpunkt in einer einzigen atomaren Operation in die Datenbank übertragen. Dazu wird auf Kopien der zu verändernden Objekte gearbeitet, die über Hilfskataloge verwaltet werden. Zum Commit-Zeitpunkt tauschen der Haupt- und der Hilfskatalog ihre Rollen. (1,5 Punkte)

Das Verfahren wird für die Realisierung der NO-UNDO/NO-REDO Strategie eingesetzt. Folglich muß auch im Falle eines Systemabsturzes nichts zur Wiederherstellung der Konsistenz geschehen, da ja nur auf Kopien gearbeitet wurde. (1,5 Punkte)

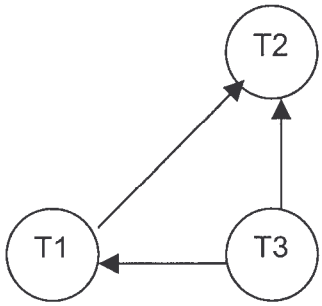
c) In diesem Fall ist die UNDO/NO-REDO Recovery-Strategie anzuweden, d.h., das Recovery muß folgendermaßen aussehen: (4 Punkte)

T1, T3 und T5 müssen nicht berücksichtigt werden, da sie zum Zeitpunkt des Systemzusammenbruchs bereits abgeschlossen waren.
T2 und T4 müssen zurückgesetzt werden (Rollback), d.h. die Before-Images müssen in die Datenbank eingebracht werden.

Aufgabe 8: Serialisierbarkeit **(6 Punkte)**

a) (3 Punkte)
Konflikte in S_1:

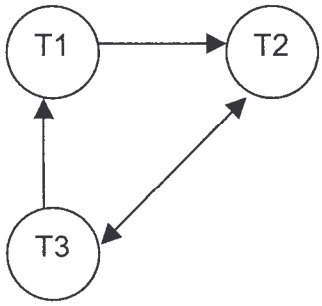
$r_3(X) - w_1(X) \rightarrow$ T3 vor T1
$r_1(Z) - w_2(Z) \rightarrow$ T1 vor T2
$w_3(Y) - r_2(Y) \rightarrow$ T3 vor T2



Der Abhängigkeitsgraph enthält keinen Zyklus. Somit ist die Schedule serialisierbar. Serielle Schedule: T3, T1, T2.

b) (3 Punkte)
Konflikte in S_2:

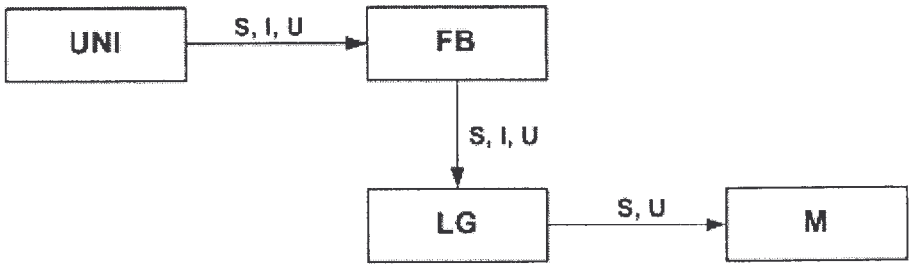
$r_3(X) - w_1(X) \rightarrow$ T3 vor T1
$r_1(Z) - w_2(Z) \rightarrow$ T1 vor T2
$r_2(Y) - w_3(Y) - w_2(Y) \rightarrow$ T2 vor T3 und T3 vor T2



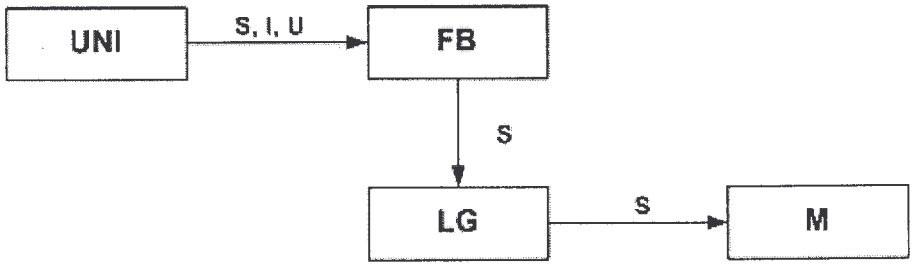
Der Abhängigkeitsgraph enthält Zyklen:
T2 → T3 → T2, aber auch
T1 → T2 → T3 → T1.
Die Schedule ist nicht serialisierbar.

Aufgabe 9: Zugriffskontrolle (15 Punkte)

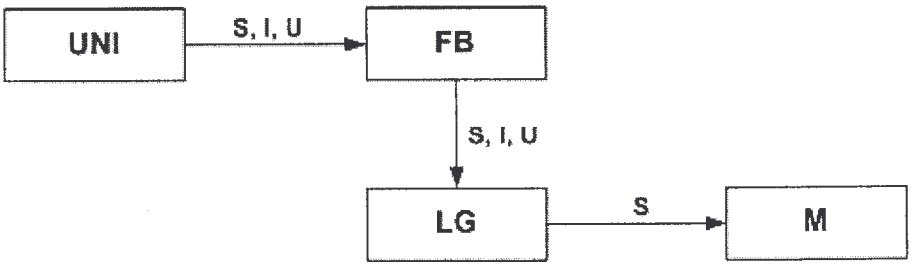
a) Abhängigkeitsgraph zur Relation VORLESUNGEN: (3 Punkte)



Abhängigkeitsgraph zur Relation PROFESSOREN: (3 Punkte)



Abhängigkeitsgraph zur Relation LESEN: (3 Punkte)



b) (je 0,5 Punkte)

FB: GRANT SELECT, INSERT, UPDATE ON VORLESUNGEN TO LG WITH GRANT OPTION
FB: GRANT SELECT ON PROFESSOREN TO LG WITH GRANT OPTION
FB: GRANT SELECT, INSERT, UPDATE ON LESEN TO LG WITH GRANT OPTION
LG: GRANT SELECT, UPDATE ON VORLESUNGEN TO M WITH GRANT OPTION
LG: GRANT SELECT ON PROFESSOREN TO M WITH GRANT OPTION
LG: GRANT SELECT ON LESEN TO M WITH GRANT OPTION

c) (je 1 Punkt)

FB: REVOKE INSERT, UPDATE ON VORLESUNGEN FROM LG
FB REVOKE INSERT, UPDATE ON LESEN FROM LG

Damit müssen die Änderungsrechte der Mitarbeiter an den Lehrgebieten zurückgenommen werde. Also:

LG: REVOKE UPDATE ON VORLESUNGEN FROM M